THE UNIVERSITY OF MICHIGAN

CONCOMP

November 1967

# PDP-8 ASSEMBLER

Michael Powers

THE UNIVERSITY OF MICHIGAN

Memorandum 12

# PDP-8 ASSEMBLER

Michael Powers

CONCOMP: Research in Conversational Use of Computers
ORA Project 07449
F.H. Westervelt, Director

supported by:

DEPARTMENT OF DEFENSE
ADVANCED RESEARCH PROJECTS AGENCY
WASHINGTON, D.C.

CONTRACT NO. DA-49-083 OSA-3050
ARPA ORDER NO. 716

administered through:

OFFICE OF RESEARCH ADMINISTRATION    ANN ARBOR

November 1967

## PREFACE

This report can be viewed as either an operating manual for the PDP-8 assembler or as a progress report on the Dexembler. The Dexembler is an assembler which ho~~pe~~fully will assemble PDP-7, PDP-8, or PDP-9 programs, depending on which of several possible tables it reads. The PDP-8 assembler, 8ASS is a realization of this assembler, but one which is not fully parameterized. As described in this report, the PDP-8 Assembler produces only absolute code and assembles only for the PDP-8 (and PDP-4).

Michael Powers
25 October 1967

TABLE OF CONTENTS

BLANK PAGE

# PDP-8 ASSEMBLER

## I. INTRODUCTION

The following sections describe the PDP-8 Assembler (8ASS), which is a collection of programs written mostly in FORTRAN IV (G) and operating under the Michigan Terminal System (MTS) on the IBM 360/67. 8ASS assembles programs for the Digital Equipment Corporation's (DEC) PDP-5 and PDP-8 computers. Once a program has been assembled, it may be punched on cards, saved in a file, or transmitted through the Data Concentrator over data lines. It is also possible to obtain binary paper tapes by use of the Data Concentrator.

The reader is assumed to be familiar with the reference manual for the PDP-8 available from DEC (Programmed Data Processor-8 User's Handbook, DIGITAL F-85, Digital Equipment Corporation, Maynard, Mass., 1964). For the description and use of assemblers in general the reader is referred to the description of the PAL-III assembler for the PDP-8 available from DEC (PAL-III Symbolic Assembler Programming Manual, DIGITAL 8/3/S, Digital Equipment Corporation, Maynard, Mass., 1965). 8ASS follows the PAL-III operation code and addressing conventions. The input format and program listing conventions of 8ASS are slightly different from those of PAL-III, however, since 8ASS is organized around a line format while PAL-III is organized around a paper tape format.

## II. ASSEMBLY PROCESSING

An assembler is a vehicle for the transformation of symbolic source programs into the internal representation of machine instructions and data. Each PDP-8 machine instruction occupies exactly one location in its memory. The assembly language program is a sequence of input lines to the assembler which specifies these instructions in symbolic form. The assembler reads these lines and constructs, or assembles, the corresponding PDP-8 binary words.

Symbolic names for the PDP-8 memory locations are defined by their appearance at the beginning of an input line. Symbolic names for operation codes appear next, sometimes followed by operands. The assembler lists a value corresponding to the value of the operator, augmented by the value of the operand. Each such value is associated with a PDP-8 address by means of the instruction location counter (ILC). The ILC contains a value which is incremented modulo 4096 after each PDP-8 word is generated. Normally, therefore, assembled words are placed in sequentially ascending locations in PDP-8 memory.

Some input lines do not generate PDP-8 words, but activate internal procedures in 8ASS. Several names which may appear in the operand are not operation codes but procedure calls (see Section 6). For example, the procedure call ORG resets the value of the ILC, allowing the programmer to control the starting location of a block of words.

The symbolic information on each assembly language line is grouped into four fields: the label, operation code (opcode), operand, and comment fields. These fields are delimited by blanks.

The label field starts at character 1 and is terminated by the first blank. If it is non-empty it may contain a name of up to eight characters, beginning with a letter. Any variable used in the program must be defined by its appearance in the label field, and the variables used with some procedure calls must be predefined, that is, defined at some point before the procedure call is processed.

The opcode field is the expression starting with the first non-blank character after the label field, and ending with the next blank. Any variable appearing in the opcode field must be an operation code.

If the operation code is a microinstruction or a self-defining expression, the operand field is empty. Otherwise, the operand field starts with the first non-blank character after the opcode field, and ends with the next blank. Any variable appearing in the operand field must be a label.

The three fields discussed above may extend to the 72nd character. The comment field starts at the end of the operand field and may extend through the 80th character. It has no effect on the binary output of the assembler—it is merely copied onto the assembly listing—but is useful to the programmer as a method of documentation. If the first character of the source line is an asterisk (*), the label, opcode, and operand field are all empty and the card is just copied onto the output listing.

There are two kinds of output from the assembler, a binary "deck" and an assembly listing. The former is a list of the machine program in a form appropriate for loading into the PDP-8 computer. The latter, the listing, not only provides the programmer and operator of the PDP-8 with what can be an invaluable guide to the operation of the program, but also indicates some types of possible programming errors.

## III. 8ASS IN MTS

The PDP-8 Assembler is available as a library file in the Michigan Terminal System (MTS). Its use is invoked by the $RUN command, with the following logical devices specified:

| | |
|---|---|
| 1 | The assembly language input lines. |
| 2 | A table of opcodes (the library file *8ØPS). |
| 6 | A tape or file (rewindable) for intermediate storage. |
| 8 | The assembly listing (output). |
| SPUNCH | The binary output (card format). |

<u>Example</u>:

```
$RUN *8ASS;1=*SOURCE* 2-8ØPS 6=-F 8=*SINK* SPUNCH=*PUNCH*
```

Due to internal size limitations, the size of program which can be assembled is limited. If a program defines

S symbols and refers to symbols R times (including uses of operation codes and procedure calls), S and R must satisfy:

$$10(S + 65) + 2R < M.$$

Three different versions of the assembler are on file, their only difference being the corresponding value of M.

For *8ASS, M=10,000.

For *8ASS20, M=20,000.

For *8ASS30, M=30,000.

## IV. NAMES AND EXPRESSIONS

A program name is a symbol which stands for a numeric value. It may stand for a self-defining value, in which case it is called a constant, or it may stand for a value which is defined elsewhere, in which case it is called a variable. A variable may be an opcode, in which case it is defined from the input table *8OPS (see Section 3) or by use of the procedure calls OPD or OPDM, or it may be label, in which case it is defined by its appearance in the label field of some input line. If this line corresponds to a PDP-8 memory location, the defined value of the label is the address of the location; if the operation field of the line is the procedure call EQU, the defined value of the label is the value of the expression in the operand field.

The special program name, *, is self-defining. Its value is the current contents of the ILC (the value "here").

The following EBCDIC characters may be used in the formation of names and expressions.

Alphabetic

upper-case letters A-Z

Numeric

digits 0-9

Operators

+ - (plus, minus)

Delimiters

| | |
|---|---|
| expression field delimiter | (blank) |
| comment field delimiter | ; (semicolon) |

Literal prefix

=

Program names must be less than nine characters long. Variables may contain alphabetic and numeric characters, but must begin with a letter. Constants must start with a digit and may contain digits and A, B, C, D, E, F (see HEXMOD, Section 5).

An _expression_ is a sequence of program names, separated by the operators + and - , and delimited by blanks. In the opcode field, any variables must be opcodes or procedure calls; in the operand field, any variables must be labels. The assembler evaluates the expression from left to right by combining the values of the names according to the operators. In the opcode field, and in the operand fields of an OPD or OPDM line, the operator + combines values by the logical OR operation. In the operand field of other procedure calls and memory-reference instructions, the values are combined arithmetically (+ for addition, - for 2's complement subtraction) modulo 4096.

An operand-field expression may be prefixed with an equal sign (=) which designates an occurrence of a literal. The value of the expression itself is termed the value of the literal, and the location to which it is assigned is termed its address. All such literal occurrences are saved in a special pool during assembler processing. When a LIT procedure call is encountered, this pool is assigned machine locations while multiple occurrences of the same value are suppressed. All literal occurrences up to this point are replaced with addresses which point to the assigned value. All symbols used in a literal expression must be predefined.

When an expression is evaluated in the operand field of a memory-reference instruction, a check is made to determine whether the value of the expression is within the current memory page. If it is, then the same-page bit of the assembled instruction is set to one. If a memory-reference instruction opcode expression is immediately followed by an asterisk, * , then the indirect bit of the assembled instruction is set to one. The I and Z conventions of PAL-III are invalid in 8ASS.

## V. INSTRUCTIONS AND PROCEDURE CALLS

A standard set of PDP-8 instruction codes is defined into the *8ASS internal symbol table from an external table such as *80PS. The opcodes in the list *80PS include the memory-reference instructions; microinstructions (Group 1 and Group 2 operate instructions, the extended arithmetic (EAE) instructions, the teletype IOT instructions); and a number of procedure calls. The machine instruction codes and their values are listed in the Appendix.

Combined microinstructions can be written as an opcode expression of microinstructions separated by + operators. This has the effect of forming the inclusive OR of the respective values. New instructions can be defined with the OPD and OPDM procedure calls.

Procedure calls are opcodes which do not represent PDP-8 machine instruction, but are signals to the assembler to invoke special procedures. The procedure calls (also known as pseudo-operations, or pseudo-ops) of *8ASS and the effects of their procedures are summarized below.

DC—define constant

Define the (optional) symbol in the label field to have a value equal to the current contents of the instruction location counter (ILC). Then substitute the value of the expression in the operand field itself for the memory location signified by the current ILC. (The DC pseudo-op provides the facility for defining decimal, octal, or address constants in a fashion parelleling the PAL-III custom of placing the name of the constant itself in the operation field.)

DECMOD—define constant conversion mode decimal
Set constant conversion to the decimal radix (normal mode is octal). May be used alternately with the OCTMOD procedure call any number of times in a program. Note: If any constant is followed by one of the letters K or T , then that constant is assumed of radix eight or ten, respectively, regardless of the current mode.

DS—define storage
Define the (optional) symbol in the label field to have a value equal to the current instruction location counter (ILC). Then add the value of the expression (predefined) in the operand field to the ILC.

END—end assembly
(Identical to the $ function of PAL-III.) Define the (optional) symbol in the label field to have a value equal to the current instruction location counter (ILC). If the operand expression is non-null, then its value will be punched on a binary transfer card as the starting address of the program.

EQU—symbolic equivalence
Define the name in the label field to have a value equal to that of the expression (predefined) in the operand field. (Similar to the = function of PAL-III.)

LIT—begin literal pool
Begin assignment of literals collected so far in the program.

OCTMOD—define constant conversion mode
Set constant conversion to the octal radix (normal mode). May be used alternately with the DECMOD procedure call any number of times in a program.

OPD—operation code definition
Define the name in the label field to designate an instruction which has an operation code equal to the value of the expression (predefined) in the operand field. (Note: The operation and symbol tables of the 8ASS assembler are disjoint so that name conflicts can be avoided. In the PAL-III assembler this is not the case: operation names used in the operand fields must be disjoint.)

OPDM—memory—reference instruction code definition
Operates identically to the OPD pseudo-op except that the operation code is presumed to designate a memory-referenced instruction.

ORG—reset instruction location counter
Reset instruction location counter (ILC) to the value of the expression (predefined) in the operand field. (Identical to the * function in PAL-III.)

## VI. DEBUGGING AIDS

When the assembler can detect an irresolvable ambiguity or inconsistency, it prints error comments on the assembly listing. Typical comments and their meanings are listed below.

"MULTIPLY DEFINED SYMBOL nnnnnnnn xxxx VARIABLE" or "...OPCODE." The name "nnnnnnnn" was defined more than once as a variable by its appearance in the label field and/or by the EQU procedure call, or more than once as an opcode by its appearance in the standard instruction table (8ØPS) and/or by the procedure call OPD or OPDM. In any case, the line is printed, with "xxxx" equal to the defined value, once for each definition. These comments are printed before the assembly listing; the four listed below are printed just before the line to which they apply. The value punched and listed for the appropriate ILC value is probably wrong.

"UNDEFINED PROGRAM NAME." During the evaluation of an expression, a name was encountered which was not defined in the program. Note that names in some procedure calls, and in literal expressions, must be predefined.

"OFF-PAGE REFERENCE." The value of the operand expression of a memory-reference instruction is neither an address on page 0 nor an address on the current page.

"INVALID OPERATOR EXPRESSION." The expression in the operator field is invalid. For example, there may be a label in the expression.

"OPERATOR-OPERAND CONFLICT." The opcodes given are incompatible, or the operator and operand are incompatible. For example, the invalid operator expression "OSR + RAR" has a Group 2 and a Group 1 opcode.

A cross-reference table is printed at the end of the assembly. It lists each variable (label or opcode) used by the program, along with its value and the contents of the ILC at each time it was used.

A summary of the number of error comments printed, the number of source lines processed, the number of symbols defined (including the standard table), the number of references to defined symbols, and the number of card images produced follows the cross-reference table.

## VII. OBJECT DECKS

8ASS produces column binary card images suitable for punching and/or loading into a PDP-8. Text cards contain numbers to be loaded into PDP-8 memory. A transfer card is punched by the END procedure call if its operand field is non-empty. The transfer card is usually used to specify a starting address for the PDP-8 program. The format of a text card is, by column:

| | |
|---|---|
| Col. 1. | a 6-7-9 punch, indicating a text card |
| Col. 2. | N , the number of contiguous PDP-8 words specified by this card ($N \leqslant 68$) |
| Col. 3. | the address of the first word in the block |
| Col. 4 | |
| ⋮ | consecutive PDP-8 word values |
| Col. 3+N | |
| Col. 4+N | a checksum, the arithmetic sum of Columns 2 through 3+N, modulo 4096. |

The format of a transfer card is:

| | |
|---|---|
| Col. 1 | a 5-7-9 punch, indicating a transfer card |
| Col. 2 | 0 |
| Col. 3 | the starting address of the program |
| Col. 4 | a checksum |

## APPENDIX I

## 8ASS STANDARD OPCODES

The following opcodes are defined as standard from the table *8θPS. The codes are octal.

I. Memory Reference Instructions

These opcodes may carry the indirect reference modifier, * , and take an operand in which any name must be a label.

| NAME | CODE |
|---|---|
| AND | 0000 |
| DCA | 3000 |
| ISZ | 2000 |
| JMP | 5000 |
| JMS | 4000 |
| TAD | 1000 |

II. Microinstructions.

A. Input-Output Instruction (IOT'S)

| NAME | CODE |
|---|---|
| IOF | 6002 |
| ION | 6001 |
| IOT | 6000 |
| KCC | 6032 |
| KRB | 6036 |
| KRS | 6034 |
| KSF | 6031 |
| TCF | 6042 |
| TLS | 6046 |
| TSF | 6041 |

B. Group I Operate Instructions

| NAME | CODE |
|---|---|
| CIA | 7041 |
| CLA | 7200 |
| CLL | 7100 |
| CMA | 7040 |
| CML | 7020 |
| GLK | 7204 |
| IAC | 7001 |
| NOP | 7000 |
| OPR | 7000 |
| RAL | 7004 |
| RAR | 7010 |
| RTL | 7006 |
| RTR | 7012 |
| STA | 7240 |
| STL | 7120 |

C. Group II Operate Instruction

| NAME | CODE | |
|---|---|---|
| CLA | 7600 | (when combined with others in Group II) |
| HLT | 7402 | |
| LAS | 7604 | |
| OSR | 7404 | |
| SKP | 7410 | |
| SNL | 7420 | |
| SMA | 7500 | |
| SNA | 7450 | |
| SPA | 7510 | |
| SZA | 7440 | |
| SZL | 7430 | |

D. Extended Arithmetic Element

| NAME | CODE | |
|---|---|---|
| ASR | 7415 | |
| CAM | 7621 | |
| CLA | 7601 | (when combined with other EAE'S) |
| DVI | 7407 | |
| LSR | 7417 | |
| MQA | 7501 | |
| MQL | 7421 | |
| MUY | 7405 | |
| NMI | 7411 | |
| SCA | 7441 | |
| SHL | 7413 | |